Submitted to *INFORMS Journal on Computing*
manuscript JOC-2017-12-OA-204

A Flexible Natural Formulation for the Network Design Problem with Vulnerability Constraints

Okan Arslan
CIRRELT and HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7, Canada, okan.arslan@hec.ca

Ola Jabali
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133 Milano, Italy, ola.jabali@polimi.it

Gilbert Laporte
CIRRELT and HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, H3T 2A7, Canada, gilbert.laporte@cirrelt.ca

Given a graph, a set of origin-destination (OD) pairs with communication requirements and an integer $k \geq 2$, the network design problem with vulnerability constraints (NDPVC) is to identify a subgraph with the minimum total edge costs such that between each OD pair, there exist a hop-constrained primary path, and a hop-constrained backup path after any $k-1$ edges of the graph fail. Formulations exist for single edge failures (i.e., $k=2$). In order to solve the NDPVC for an arbitrary number of edge failures, we develop two natural formulations based on the notion of length-bounded cuts. We compare their strengths and flexibilities in solving the problem for $k \geq 3$. We study different methods to separate infeasible solutions by computing length-bounded cuts of a given size. Experimental results show that for single edge failures, our formulation increases the number of solved benchmark instances from 61% (obtained within a two-hour limit by the best published algorithm) to over 95%, thus increasing the number of solved instances by 1065. Our formulation also accelerates the solution process for larger hop limits and efficiently solves the NDPVC for general k. We test our best algorithm for two to five simultaneous edge failures and investigate the impact of multiple failures on the network design.

Key words: network design; survivability; vulnerability; branch-and-cut algorithm; length-bounded minimum cut

History:

1. Introduction

The purpose of this paper is to develop exact solution algorithms for the network design problem with vulnerability constraints (NDPVC), capable of handling a general number of edge failures. To formally define the problem, consider a graph $G=(N,E)$, where N is the set of nodes, and E is the set of edges, with edge cost c_e for $e \in E$. The vulnerability of

the graph to failures is determined by an integer parameter $k \geq 2$, equal to the maximum number of simultaneously failing edges in the graph, plus one. We seek to find two types of paths, 'primary' and 'backup', for a set of origin and destination (OD) pairs. The primary paths are used to connect the OD pairs when there are no failures in the graph and the backup paths are used when the failures obstruct the primary paths. A demand r is defined as a quadruple (s_r, t_r, H_r^p, H_r^b), where s_r and t_r are the origin and destination nodes, respectively, and H_r^p and H_r^b are the hop bounds on the primary and backup paths, respectively. We use length of a path or number of hops interchangeably to refer to the number of edges used in the path. Let $\mathcal{R}$ be the set of all demands. For a given NDPVC instance $(G, \mathcal{R}, k)$, the problem is defined as identifying a subgraph of G with the minimum total edge costs such that, between nodes s_r and t_r for $r \in \mathcal{R}$, there exists a primary path of length at most H_r^p edges, and a backup path of length at most H_r^b edges after any $k-1$ edges of the graph fail. The NDPVC was first introduced by Gouveia and Leitner (2017a) and different mixed-integer linear programming (MILP) models were developed for a single edge failure (i.e., for $k = 2$). Their flow-based models are built on the idea of finding a set of 'backup edges' for each OD pair in order to maintain the communication after the edges on the primary path fail. The authors then improved on their previous work by developing branch-and-cut and Benders decomposition algorithms (Gouveia et al. 2018). In this paper, we formulate and solve the NDPVC for $k \geq 2$. The case of $k = 2$ is equivalent to the problem studied by Gouveia et al. (2018), for which we substantially accelerated the solution process. Furthermore, we solved the problem for $k \geq 3$, i.e., considering multiple edge failures.

The NDPVC is closely related to the survivable network design problem, in particular to the k-edge-disjoint network design problem (k-ESNDP), defined as identifying a subgraph of G with the minimum total edge costs, such that for each OD pair, there exists k edge-disjoint paths. Stoer (1992) and Grötschel et al. (1995) investigated the polyhedral properties of the problem for $k = 2$ and general k, respectively. For a comprehensive overview of the k-HSNDP, we refer the reader to Kerivin and Mahjoub (2005), which covers properties, polynomially solvable special cases, polyhedral analysis and cutting plane approaches. We also refer the reader to Bendali et al. (2010) for a more recent reference on cutting plane methods for the k-edge-connected subgraph problem. Note that a solution of the k-ESNDP can result in very long paths. For example, a Hamiltonian cycle is a feasible

solution in which all OD pairs are connected by two different paths. To account for the quality of service in transportation and telecommunications networks, it is common to take into account the number of edges on the path, often referred to as 'hop bound' (Balakrishnan and Altinkemer 1992). Fortz et al. (2000), Fortz and Labbé (2002), Fortz et al. (2006) investigated the survivability in networks by considering hop-constrained cycles connecting OD pairs by two alternative paths. Gouveia (1996, 1998) considered hop constraints in spanning trees. The hop-constrained survivable network design problem (k-HSNDP) is an extension of k-ESNDP obtained by considering a hop constraint for each OD pair. Building on the layered network flow formulation of Gouveia (1998), Botton et al. (2013) presented a Benders decomposition algorithm for the k-HSNDP for arbitrary k and length bounds.

The main difference between the k-HSNDP and the NDPVC is that in the former problem, edge-disjoint paths are constructed, whereas in the latter problem, the primary and backup paths need not be disjoint, the only requirement being to ensure hop-constrained connectivity of sources to their destinations after the failure of any k edges in the graph. Hence, the k-HSNDP solution is more conservative and can result in suboptimal or infeasible solutions to instances for which NDPVC has solutions with arbitrarily smaller total costs (Gouveia and Leitner 2017a).

Botton et al. (2013) refer to the models that use only a single variable for each edge and an exponential number of constraints as 'natural models' and argue that the best computational performances are generally obtained through these models. The authors also discuss the difficulties associated with finding such models when hop bounds are present in the problem. In previous NDPVC formulations developed by Gouveia and Leitner (2017a) and Gouveia et al. (2018), the authors rely on variables for each combination of edge, OD pair and hop bound. Therefore, the models are prone to stalling for large problem instances. In contrast natural formulations would not suffer from increasing the number of variables for large networks but, increasing network size can have a significant impact on the number of needed constraints. As a result, it is critical to develop efficient separation techniques in order to make such models work efficiently.

1.1. Scientific Contributions

In this paper, we develop two natural models with exponential number of constraints for the NDPVC with an arbitrary number of edge failures. We compare the two models in terms of their strengths and flexibility in solving the problem when multiple simultaneous

failing edges are present. We develop an exact branch-and-cut algorithm to solve these models. The separation problem consists of identifying a length bounded cut of a given size. We develop a model and an algorithm to solve the separation problem, and investigate the properties of the two NDPVC models to accelerate the implementation. Our best model solves 95% of the benchmark instances for single edge failures, of which only 61% could be solved by the best published algorithm (Gouveia et al. 2018). We also show that our model performs well when the hop bound is increased. Furthermore, our best model can be used to solve the problem for $k \geq 3$, which was not yet solved in the literature. We report computational results for $k = 2, \ldots, 6$.

1.2. Organization of This Paper

The remainder of this paper is organized as follows. In Section 2, we develop our models and investigate their properties. Both of our formulations depend on the notion of length-bounded cuts. Section 3 presents alternative ways of solving the length-bounded cut problem, which arises as the separation problem for our formulations. In Section 4, we present the data, the experimental settings, implementation details and computational results, and we conclude the study in Section 5.

2. Mathematical Models

We first introduce our notation, and we then develop two alternative natural models and compare their strengths and flexibilities in solving the NDPVC for $k \geq 3$.

2.1. Notation

Consider the undirected graph $G = (N, E)$, where N is the set of nodes and E is the set of edges $[i, j]$ with $i, j \in N$ and by convention $i < j$. We also define an associated directed graph $\bar{G} = (N, A)$ where the arc set A contains two opposite arcs for each edge $[i, j] \in E$. Let d_{ij} be the number of minimum hops, equivalently the 'distance', in $\bar{G}$ between nodes $i, j \in N$. We say that edge $[i, j]$ *corresponds to* arcs (i, j) and (j, i), and vice versa. We define $A_r^p = \{(i, j) \in A : d_{s_r i} + d_{j t_r} + 1 \leq H_r^p\}$ and $A_r^b = \{(i, j) \in A : d_{s_r i} + d_{j t_r} + 1 \leq H_r^b\}$. Let N_r^p and N_r^b be the node sets induced by arcs in A_r^p and A_r^b, respectively. We refer to the graphs $G_r^p = (N_r^p, A_r^p)$ and $G_r^b = (N_r^b, A_r^b)$ as the primary and backup graphs, respectively, of demand $r \in \mathcal{R}$.

We also recall the definition of a *length bounded cut (lb-cut)*. Given $\bar{G} = (N, A)$, a positive integer H as a length bound on the paths and an OD pair (s, t), a set of arcs $\bar{S} \subset A$ is called

an lb-cut, if the removal of the arcs of $\bar{S}$ destroys all paths of length at most H from s to t. Observe that there can be more lb-cuts in $\bar{G}$ than the number of cuts that destroy all paths between the same OD pair. Let $S \subset E$ be the set of *edges* corresponding to the arcs in $\bar{S} \subset A$. We also refer to edge set S as an lb-cut, since it destroys all undirected paths between the same OD pair in the undirected graph G. Let Γ_r^p be the set of all such edge sets S corresponding to the lb-cuts $\bar{S}$ of length bound H_r^p in G_r^p. Similarly Γ_r^b is defined as the set of all edge sets $S \subset E$ corresponding to the lb-cuts $\bar{S} \subset A$ of length bound H_r^p in G_r^b.

For an edge set $\Lambda \subset E$, let $G_r^b(\Lambda)$ be the graph induced by the arcs $A_r^b(\Lambda) := \{(i,j) \in A_r^b : [i,j] \notin \Lambda\}$. We also define $\Gamma_r^b(\Lambda)$ as the set of all edge sets $S \subset E$ corresponding to the lb-cuts $\bar{S} \subset A_r^b(\Lambda)$ of length bound H_r^p in $G_r^b(\Lambda)$. Finally, for convenience, we refer to $\sum_{e \in S} x_e$ as $x(S)$.

2.2. A Natural Formulation

We now develop a natural formulation for the NDPVC, which we refer to as NF. In our formulation, we ensure the existence of a hop-constrained path using lb-cuts. The lb-cuts are also known as jump inequalities, see, e.g. Dahl et al. (2006), where they are used to model the hop-constrained minimum spanning tree problem. The same modeling technique to ours is also used by Arslan et al. (2017).

PROPOSITION 1. *For a given graph G, an OD pair (s,t) and a hop bound H, there exists a path of length at most H from s to t if and only if every lb-cut contains at least one edge of the path.*

Proof Assume that there exists a path p of length at most H from s to t. This implies that every lb-cut, by definition, includes an arc from the path p. By contraposition, assume that all paths have a length strictly greater than H. Then $\varnothing$ is an lb-cut, implying that there exists an lb-cut that contains no edge in path p. □

The NF formulation is based on the characteristic observed by Gouveia and Leitner (2017a) which entails that if a set of edges fail, then the remaining edges should enable a hop-constrained backup path. Let binary variables x_e be equal to one if and only if edge $e \in E$ belongs to the solution.

$$\text{(NF) minimize} \sum_{e \in E} c_e x_e \tag{1}$$

subject to

$$x(S) \geq 1 \qquad S \in \Gamma_r^p, r \in \mathcal{R} \tag{2}$$

$$x(S) \geq 1 \qquad S \in \Gamma_r^b(\Lambda), \Lambda \subset A_r^b : |\Lambda| = k-1, r \in \mathcal{R} \tag{3}$$

$$x_e \in \{0,1\} \qquad e \in E. \tag{4}$$

The objective function minimizes the total cost of all edges belonging to the solution. Constraints (2) ensure for every demand $r \in \mathcal{R}$ that there exists a primary path of length at most H_r^p. Constraints (3) impose, for each demand r, the existence of a backup path of length at most H_r^b when any edge set $\Lambda \subset A_r^b : |\Lambda| = k-1$ fails. Note that, for convenience, we prefer to state Constraints (2) and (3) separately rather than putting them as a single set of constraints. Constraints (4) are the integrality requirements. Since the number of constraints (2) and (3) is exponential, we will develop a branch-and-cut algorithm to solve NF. The details of separation algorithms are presented in Section 3.

2.3. A Flexible Formulation

Even though NF can handle an arbitrary number of edge failures, it requires the enumeration of all subsets of edges set of cardinality $k-1$. Therefore, identifying violation becomes cumbersome and the separation problem fast becomes difficult to solve for increasing k values. In order to develop a more flexible model for the general case with an arbitrary number of edge failures, we need the following theorem, which is one of the main contributions of this paper.

THEOREM 1. *For given $r \in \mathcal{R}$, an integer $k \geq 2$ and a binary vector $x \in \{0,1\}^{|E|}$, there exists a hop-constrained backup path of length at most H_r^b after failure of any $k-1$ edges in the graph, if and only if $x(S) \geq k$ for all lb-cuts $S \in \Gamma_r^b$.*

Proof (*Sufficiency*) Assume that there exists a hop-constrained path of length at most H_r^b from s_r to t_r for $r \in \mathcal{R}$ after any $k-1$ edges fail. Let $S_{min} = \arg\min_{S \in \Gamma_r^b} \{x(S)\}$. Suppose that $f \geq 0$ failing edges are in S_{min}. By assumption, there exists a backup path of length at most H_r^b after the edges fail. By definition, the lb-cut S_{min} contains at least one of the backup path edges. Then, S_{min} contains f failing edges and at least one non-failing edge, which implies that $x(S_{min}) \geq f+1$ for all possible values of f. Since $k-1$ edges fail, we have $f \leq k-1$. The tightest condition is when $f = k-1$ and therefore $x(S_{min}) \geq k$. By definition of S_{min}, we have $x(S) \geq x(S_{min})$ for all $S \in \Gamma_r^b$. Therefore $x(S) \geq k$ for all $S \in \Gamma_r^b$.

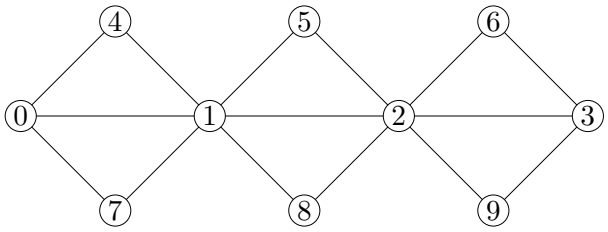

Figure 1 **An example graph with a single demand $r = (0, 3, 3, 5)$ and $k = 3$.**

(*Necessity*) Similar to the proof of Proposition 1: by contraposition, suppose that the length of every path is strictly greater than H_r^b after failure of edges; then, $\varnothing$ is an lb-cut with $x(\varnothing) < k$. □

The following corollary shows that having $x(S) \geq k$ with vector $x \in \mathbb{R}^{|E|}$ and $0 \leq x_e \leq 1$ ensures the existence of a hop-constrained path.

COROLLARY 1. *For a given $r \in \mathcal{R}$, an integer $k \geq 2$ and a vector $x \in \mathbb{R}^{|E|}$ with $0 \leq x_e \leq 1$ for all $e \in E$, if $x(S) \geq k$ for all lb-cuts $S \in \Gamma_r^b$, then there exists a hop-constrained backup path of length at most H_r^b after failure of any $k-1$ edges in graph G_r^b.*

Proof $x(S) \geq k$ and $x_e \leq 1$ for all $e \in E$ implies that for every lb-cut $S \in \Gamma_r^b$, at least k edges satisfy $x_e > 0$ for all $e \in S$. Then, from Theorem 1 there exists a hop-constrained path of length at most H_r^b in the graph induced by edges with $x_e > 0$ after failure of any $k-1$ edges. □

An example of a graph with a single demand r with $s_r = 0$, $t_r = 3$, $H_r^p = 3$, $H_r^b = 5$ and $k = 3$ is depicted in Figure 1. The only feasible primary path is $(0, 1, 2, 3)$. Note that if the two edges on the primary path fail (e.g., $[0, 1]$ and $[1, 2]$), then having at least one alternative path, e.g., $(0, 4, 1, 5, 2, 3)$ suffices to ensure the existence of a backup path. However, when at least one edge which is not on the primary path fails (e.g., edge $[0, 4]$ not on the primary path and edge $[0, 1]$ on the primary path), then the cardinality of each length-bounded cut must be at least three. Note that in the above example, the solution to NDPVC contains all edges.

Theorem 1 provides us with constraints to model the backup paths without enumerating any edge set, as is done in Constraints (3). Put differently, there exists a more flexible way to model the backup paths. We shall therefore refer to our new model as the 'Flexible Formulation' (FF):

$$\text{(FF) minimize } \sum_{e \in E} c_e x_e$$

subject to (2), (4)

$$x(S) \geq k \qquad\qquad S \in \Gamma_r^b, r \in \mathcal{R}. \tag{5}$$

The difference between NF and FF is that Constraints (3) are replaced by Constraints (5), which ensure the existence of a hop-constrained backup path after the failure of any $k-1$ edges in G, due to Theorem 1. Note that in contrast to NF, FF does not require the enumeration of edge subsets.

2.4. Comparison of the Two Formulations

LEMMA 1. *For $r \in \mathcal{R}$ and an integer $k \geq 2$, consider the graph G_r^b. Let $\Lambda \subset A_r^b$ be a set of edges with $|\Lambda| = k-1$ and $S \in \Gamma_r^b$ be an lb-cut in G_r^b. Then $S \setminus \Lambda$ is an lb-cut in $G_r^b(\Lambda)$.*

Proof Let $\mathcal{P}(\Lambda)$ be the set of length-bounded paths in G_r^b having at least one edge in the set Λ, and let $\overline{\mathcal{P}}(\Lambda)$ be the set of length-bounded paths in G_r^b with no common edge with the set Λ. Since the cut S destroys all paths in $\mathcal{P}(\Lambda) \cup \overline{\mathcal{P}}(\Lambda)$ and Λ destroys all paths in $\mathcal{P}(\Lambda)$, we have the desired result. □

Let $\mathcal{X}_{\text{NF}}$ and $\mathcal{X}_{\text{FF}}$ be the feasible sets of the NF and FF formulations, respectively.

PROPOSITION 2. *$\mathcal{X}_{FF} \subseteq \mathcal{X}_{NF}$.*

Proof We investigate the only difference in formulations, which is Constraints (3) being replaced by Constraints (5). Let $x^* \in \mathcal{X}_{FF}$, $S \in \Gamma_{\hat{r}}^b$ for $\hat{r} \in \mathcal{R}$ and $\Lambda \subset S$ with $|\Lambda| = k-1$. Then $x(S) \geq k$, which can be rewritten as $x(\Lambda) + x(S \setminus \Lambda) \geq k$. Since $x(\Lambda) \leq k-1$, it follows that $x(S \setminus \Lambda) \geq 1$. Due to Lemma 1, $S \setminus \Lambda$ is an lb-cut in $G_r^b(\Lambda)$. Therefore, $x^* \in NF$. □

To show that the inclusion is strict, consider a single demand with three length-bounded and edge-disjoint paths connecting its origin to its destination and $k = 2$. A solution with all variables corresponding to edges equal to 0.5 satisfies Constraints (3), but not Constraints (5).

3. Separation Problems and Algorithms

Given a solution $x^* \in \mathbb{R}^E$ of NDPVC, the separation problem for Constraints (2), (3) and (5) is to identify an lb-cut of a given weight or to conclude that none exists. For $r \in \mathcal{R}$, we separate

- Constraints (2) by an lb-cut $S \in \Gamma_r^p$ with $x^*(S) < 1$,
- Constraints (3) by an lb-cut $S \in \Gamma_r^b(\Lambda), \Lambda \subset A_r^b : |\Lambda| = k-1$ with $x^*(S) < 1$ and

- Constraints (5) by an lb-cut $S \in \Gamma_r^b$ with $x^*(S) < k$.

In what follows, we provide some background on lb-cuts, as well as alternative techniques to separate these inequalities. Menger's Theorem (Menger 1927) states that in a graph G, the maximum number of edge-disjoint paths between two nodes s and t equals the minimum cardinality s-t cut. Ford and Fulkerson (1956) generalized Menger's result by showing that the maximum flow between nodes s and t equals the minimum capacity s-t cut. In particular, this cut destroys *all* s-t paths. The search for lb-cuts dates back to the work of Adámek and Koubek (1971) who showed that the Ford and Fulkerson's max flow-min cut theorem may not hold when bounds are imposed on the path lengths. These authors also argued that for length bounds of at most three, the max flow equals the min cut. Early works on bounded paths include that of Lovász et al. (1978), in which the Mengerian theorems for paths of bounded length are discussed. Mahjoub and McCormick (2010) presented a graph transformation, which we discuss in Section 3.1, to find the minimum lb-cut for length bounds of at most three. Finding a minimum cut on their transformed graph is equivalent to finding a minimum lb-cut in the original graph. For length bounds of size at most four, it is known that finding the minimum cardinality lb-cut is NP-hard (Baier et al. 2006).

In the aforementioned studies, the objective is to minimize the cardinality of the cut. However, in our separation problem, we are concerned with the weight of an lb-cut. To this end, we present an IP model in Section 3.2, due to Arslan et al. (2017), for the solution of the minimum weight lb-cut problem. Observe that our separation problem can be solved by the decision version of the minimum lb-cut problem, when parametrized by the size k of the cut. Golovach and Thilikos (2011) presented an FTP algorithm for finding an lb-cut of cardinality at most k or for concluding that no such lb-cut exists. In Section 3.3, we adapt their algorithm to solve our separation problem. We discuss how we can use the regular minimum cut finding algorithms for our separation purposes in Section 3.4. Finally, in Section 3.5, we present a way to strengthen the generated cuts.

3.1. Length-bounded Cuts of Length at Most Three

Mahjoub and McCormick (2010) observed that all the nodes on paths of length at most three are either connected to the origin or to the destination nodes. Building on this observation, the authors developed a linear-time network transformation procedure where the maximum flow-minimum cut theorem applies to the lb-cuts in the transformed graph.

Therefore, finding a minimum cut on this transformed graph boils down to solving a minimum cut problem. We refer to this technique as *lbcut3*.

3.2. IP Model

We present an integer programming model for the identification of a minimum lb-cut, which is the arc version of the model developed by Arslan et al. (2017) for finding minimum weight length-bounded node-cuts. Given a directed graph $\hat{G} = (\hat{N}, \hat{A})$ with node and arc sets $\hat{N}$ and $\hat{A}$, respectively, a positive integer length bound H, and origin and destination nodes $s, t \in \hat{N}$. Each arc $(i, j) \in \hat{A}$ has a unit length and a non-negative weight y_{ij}. $\hat{G}$ equals G_r^p when finding lb-cuts in the primal graph to separate Constraints (2) and it equals G_r^b when finding lb-cuts in the backup graph to separate Constraints (3) and (5). Given a fractional or integer solution x^*, we obtain the arc weights y_{ij} from the corresponding edge variable values in the solution x^*. We now need the following definition.

DEFINITION 1. For a given graph $\hat{G} = (\hat{N}, \hat{A})$, an edge set $S \subset \hat{A}$ and two nodes $i, j \in \hat{N}$, the *unintercepted shortest path* is defined as the shortest path from i to j in the subgraph induced by arcs $\hat{A} \setminus S$.

Let M be a sufficiently large number, u_{ij} be an indicator variable equal to one if and only if arc $(i, j) \in \hat{A}$ is in the lb-cut and π_i be the length of the unintercepted shortest path from node $i \in \hat{N}$ to node t. Having $\pi_i \geq M$ implies that there exists no path from i to t. The following model, which we refer to as lb-cut model (lbcutM), finds a minimum-weight lb-cut in $\hat{G}$. The logic behind the model is to select a subset of arcs with the minimum weight so that the length of the unintercepted shortest path from s to t is greater than the length bound H. Put differently, no path from s to t of length at most H remains after the selected edges are removed from the graph.

$$\text{(lbcutM)} \quad \text{minimize} \sum_{(i,j)\in\hat{A}} y_{ij} u_{ij} \tag{6}$$

subject to

$$\pi_t = 0 \tag{7}$$

$$\pi_i \leq \pi_j + 1 + M u_{ij} \qquad (i, j) \in \hat{A} \tag{8}$$

$$\pi_s \geq H + 1 \tag{9}$$

$$\pi_i \geq 0 \qquad i \in \hat{N} \tag{10}$$

$$u_{ij} \in \{0, 1\} \qquad (i, j) \in A. \tag{11}$$

The objective function minimizes total weight of the lb-cut. When the model is solved, the minimum lb-cut is identified by the u variables. Constraint (7) fixes the node label π_t at destination equal to zero. Constraints (8) ensures that π_i variable is at most the length of the unintercepted shortest path from node i to node t. Constraints (9) imposes that the unintercepted shortest path length from the origin to the destination is greater than l. Constraints (10) and (11) are non-negativity and integrality requirements.

3.3. An Algorithm for Finding Length-bounded Cuts

Given a directed graph, Golovach and Thilikos (2011) developed an algorithm to find an lb-cut of at most a given size, or to conclude that no such cut exists. In this section, we present a minor modification of their algorithm so that the output is the weight of the lb-cut rather than its size. Given a directed graph $\hat{G} = (\hat{N}, \hat{A})$, an OD pair (s, t), a fractional number k, an integer H, a set of arcs $X \subset \hat{A}$ and a weight vector $w^{|\hat{A}|}$, our algorithm finds an s-t lb-cut S in $\hat{G}$, if one exists, containing the set X, such that all paths of length not exceeding H are destroyed and the total weight of the edges in cut S is strictly less than k. We refer to the algorithm presented in Algorithm-1 as lbcutA. For conciseness, let $w(S) = \sum_{(i,j)\in S} w_{ij}$. The algorithm starts by comparing the weight of the lb-cut X with the value of k and terminates if the weight is at least k. If not, a shortest s-t path p that does not include any arc from set X is identified. If its length is at least $H + 1$, then X is a feasible lb-cut of a total weight less than k. Otherwise, by definition, at least one arc on path p needs to be in the cut. The correctness of the algorithm follows from the observation that if there exists a path of length at most H, then at least one arc on this path needs to be in the lb-cut. The arguments of Golovach and Thilikos (2011) for the proof of correctness directly apply to the lbcutA algorithm. We then branch on every arc on the path p and see whether appending it to the set X makes this set a feasible lb-cut of weight less than k. With this idea, the recursive Step 7 appends a single arc at each call. Due to finiteness of the arc set, the algorithm converges. Note that the only modification to the Golovach and Thilikos (2011) algorithm is in Line 1 where we compare the weight of the cut rather than the size.

Observe that a single call of $lbcutA(\hat{G}, s, t, k, H, \emptyset, w)$ solves the separation problem for the OD pair (s, t). Furthermore, since the zero-weight arcs do not contribute to the total cut weight, calling $lbcutA(\hat{G}, s, t, k, H, X^0, w)$, where X^0 is the set of zero-weight arcs, also yields the minimum weight lb-cut and is computationally more efficient, especially in cases where

the network has many zero-weight arcs. Even though the algorithm has an exponential run time, in practice it performs well as shown in Section 4.

Algorithm 1: lbcutA$(\hat{G}, s, t, k, H, X, w)$

Input: $\hat{G}, s, t, k, H, X, w$.

Output: An s-t lb-cut $S \supseteq X$ of weight$<k$, destroying paths of length H or *false* if no such lb-cut exists.

1 **if** $w(X) \geq k$ **then return** *false*;
2 Let p be a shortest s-t path in $\hat{G} \setminus X$;
3 **if** $|p| \geq H + 1$ **then**
4 return X
5 **else**
6 **foreach** $(i, j) \in p$ **do**
7 **Set** $Y =$ lbcutA$(\hat{G}, s, t, k, l, X \cup (i, j), w)$
8 **if** $Y \neq$ *false* **then return** Y;
9 **return** *false*

Note that Constraints (2), (3) and (5) can be separated for both fractional and integer solutions using the lbcutA algorithm. Observe that when the solution is integer, we can determine whether there exists a violation of Constraints (2) by solving a single shortest path problem in line 2. This special case of our algorithm, running in polynomial time, was also presented by Arslan et al. (2017). With a similar logic, we can separate Constraints (3) for integer solutions by solving a shortest path problem for every subset of the edge set of size equal to $k-1$. This means that when $k = 2$, we only solve as many shortest path problem instances as the number of arcs in $\hat{G}$. The major issue with the separation of Constraints (3) using the lbcutA algorithm is that in order to determine any violation for $r \in \mathcal{R}$, we need to enumerate as many as $\binom{|\hat{A}|}{k-1}$ edge sets and solve a shortest path problem for each of them. This enumeration is obviously only possible for small k values. Note that there is no need for such an enumeration to separate Constraints (5) in the flexible formulation FF.

3.4. ε-Minimum Cut Heuristic

Observe that all minimum weight cuts in a graph are also lb-cuts. Therefore, any classical minimum cut finding algorithm can be used as a heuristic to identify an lb-cut and to separate both integer and fractional solutions. To strengthen a cut of type $x(S) \geq k$, we

need to shrink the size of the lb-cut S. In order to strengthen the generated cut by the heuristic, we assign a small value ε to those arcs with zero weight. This does not add significant computational burden to the algorithm, but helps to generate stronger cuts. This strengthening idea of the cuts generated by the minimum cut algorithm was implemented by Koch and Martin (1998).

3.5. Strengthening the Cuts

We strengthen the cut generated by the ε-mincut algorithm by assigning an ε value to zero-weight arcs. However, for the lbcutM model and the lbcutA algorithm, assigning ε to zero-cost arcs proved to be costly. Therefore, we first generate a minimum weight lb-cut using the lbcutM model or the lbcutA algorithm. The cut generated generally includes many zero-weight edges since these edges do not contribute to the objective function. We then modify the lbcutM model by making the objective function coefficients of the arc variables in the generated lb-cut equal to one, and equal to a large number to the remaining arc variable coefficients. This model then minimizes the size of the lb-cut. This minimization of the cuts proved to be very effective in our experiments. In order not to spend too much time to prove the optimality of a feasible solution, we set a time limit on the minimization of the cut size. We refer to this technique as min-lbcutM-h(t), where t is the time limit.

4. Computational Study

We now present the data, some implementation details, the experimental settings to test the efficiency of our models and algorithms, and the results.

4.1. Data

We used the same set of problem instances, generated for the NDPVC by Gouveia and Leitner (2017a), which is available online (see Gouveia and Leitner (2017b)). This set contains two classes of instances, referred to as *grid* and *random*. The former are based on grid graphs with chords and it consists of two subsets C and D. These subsets represent different ways to select commodities. The costs of horizontal and vertical edges are random integers between 1 and 10 and the cost of diagonal edges are random integers between 10 and 50. The random instances, on the other hand, are based on randomly generated graphs on a plane and contains two subsets, referred to here as E and R. The cost of each edge is a multiple of the Euclidean length of the edge. Further details on the generation schemes and

selection rules are discussed in the paper by Gouveia and Leitner (2017a). Each graph is associated with a set of instances as shown in Table 1. The first column provides the name of the set. The number of nodes, arcs and the demands are given in columns two through five, respectively. '#' is the number of instances in the set. Associated with every instance is a parameter H_{min} defined as the minimum number of hops required to have at least one path between each OD pair of the demands set. The three leftmost columns provide the minimum, average and maximum H_{min} values of a given instance set, respectively.

4.2. Implementation Details

Here, we present the implementation details of the tests performed to evaluate the efficiency of the different separation algorithms. We have implemented our models and algorithms using Java under Linux and CPLEX 12.7.1. All experiments were conducted on a cluster of 27 machines each having two Intel(R) Xeon(R) X5675 3.07 GHz processors running on Linux. Each machine has 12 cores and each experiment was run using a single thread. Similar to previous studies, we set the time limit for all experiments to 7200 seconds, and the memory limit to 3GB.

For a length bound $H \leq 3$, we used the lbcut3 algorithm for the exact separation, presented in Section 3.1, which requires a graph transformation and running a minimum cut algorithm. This algorithm is computationally very effective, therefore we only use lbcut3 for $H \leq 3$. For $H > 3$, we considered three ways of separating an infeasible solution: the ε-mincut algorithm, solving the lbcutM model, and the lbcutA algorithm. The ε-mincut algorithm is a heuristic, and the last two are exact separation techniques. Note that we can also run the lbcutM model and the lbcutA algorithm as heuristics by enforcing a time limit. We refer to these heuristics as lbcutM-h(t) and lbcutA-h(t), respectively, where t is the time limit in seconds.

In our preliminary experiments, we observed several instances getting stuck at the root node, not improving the lower bound significantly for many iterations. To prevent this phenomenon, we implemented a *tailing-off* rule (Sherali and Driscoll 2000). If the improvement of the best bound is not significant in the last two iterations at the root node, we then resort to branching rather than spend more time in the corner of the polytope where the algorithm is stuck.

If we cannot identify a violation with our heuristic methods, we then resort to one of our exact separation techniques: lbcutM or lbcutA. Since the separation problem itself is

Table 1 Properties of instance sets

					H_{min}		
Set	$\|N\|$	$\|A\|$	$\|\mathcal{R}\|$	#	min	avg	max
C-1	100	342	5	20	3	4.7	7
C-2	100	342	10	20	4	5.4	7
C-3	400	1482	5	20	3	4.7	7
C-4	400	1482	10	20	4	5.05	7
C-5	400	1482	20	20	4	5.75	7
C-6	900	3422	5	20	3	4.65	7
C-7	900	3422	10	20	4	5.4	7
C-8	900	3422	20	20	4	5.6	7
C-9	900	3422	30	20	4	5.9	7
D-1	25	72	10	10	3	3.8	4
D-2	49	156	10	10	4	5.2	6
D-3	100	342	10	10	7	8.2	9
D-4	100	342	45	10	6	8.1	9
D-5	400	1482	10	10	12	14.6	18
E-1	50	122	10	5	6	7.6	9
E-2	50	122	45	5	7	8.6	11
E-3	50	245	10	5	4	4.6	6
E-4	50	245	45	5	4	4.8	6
E-5	75	277	10	5	5	5.6	6
E-6	75	277	45	5	6	8.6	12
E-7	75	555	10	5	3	4.4	6
E-8	75	555	45	5	4	4.6	5
E-9	100	495	10	5	5	5.2	6
E-10	100	495	45	5	6	7.2	9
E-11	100	990	10	5	3	4.0	5
E-12	100	990	45	5	3	4.8	6
R-1	50	122	10	5	3	4.6	6
R-2	50	122	45	5	4	5.2	6
R-3	50	245	10	5	2	2.8	3
R-4	50	245	45	5	3	3.0	3
R-5	75	277	10	5	3	3.8	4
R-6	75	277	45	5	4	4.2	5
R-7	75	555	10	5	2	2.6	3
R-8	75	555	45	5	2	2.8	3
R-9	100	495	10	5	3	3.6	5
R-10	100	495	45	5	4	4.0	4
R-11	100	990	10	5	2	2.0	2
R-12	100	990	45	5	3	3.0	3

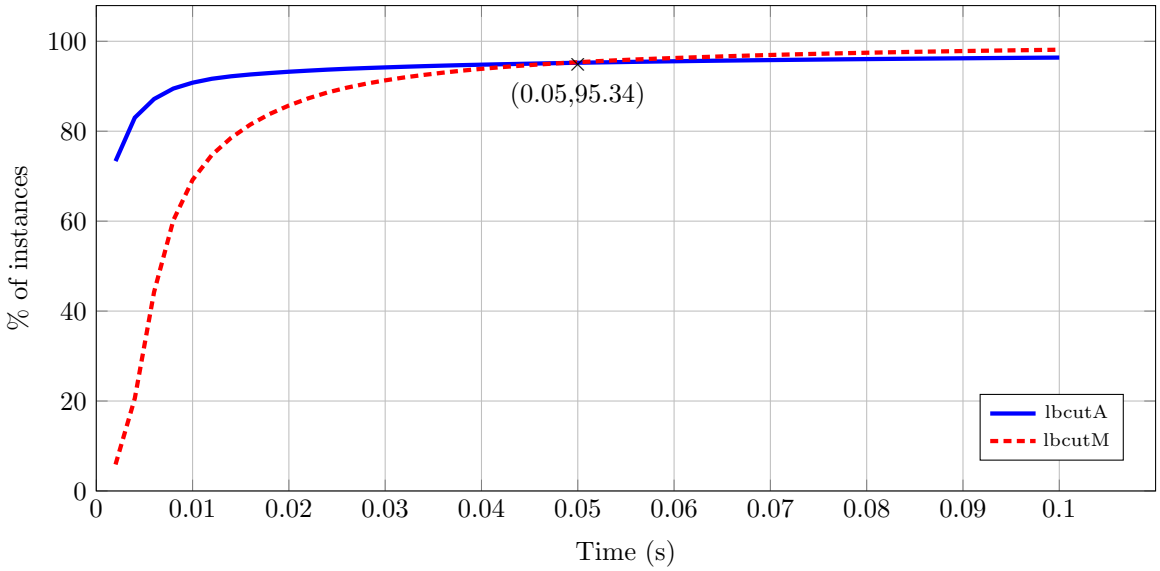

Figure 2 **Percentage of separation problems solved**

NP-hard, solution can potentially take a very long time. Therefore, rather than searching for violations in all OD pairs exactly, we search until we detect a single violation. When such a violation is identified, we add a cut, stop the search process for further violations in the remaining set of OD pairs and use the generated cut to separate the solution at hand. We refer to this technique as *quick-exit*.

We used all separation algorithms for both integer and fractional solutions. We separated fractional solutions only at the root node. We also implemented very aggressive probing (Savelsbergh 1994) at the root node.

4.3. Hybrid heuristic

To better understand the impact of each solution technique, we ran over a million separation problems on a randomly selected subset of instances from all four C, D, E and R sets. Figure 2 plots the cumulative percentage of separation instances solved versus time. We observed that the solution of the lbcutM model and the lbcutA algorithm show different progress in time for the same separation problem instance. The lbcutA algorithm performs well on the majority of instances and can solve more than 90% of them within 0.01 second. However it performs very poorly on the remaining instances: 99.87% were solved within 60 seconds (not shown in the figure) and the last 0.12% took longer times, reaching over an hour. The performance of the lbcutM model on the other hand is inferior to that of the lbcutA algorithm on 95% of the instances. However it can solve all instances within 15

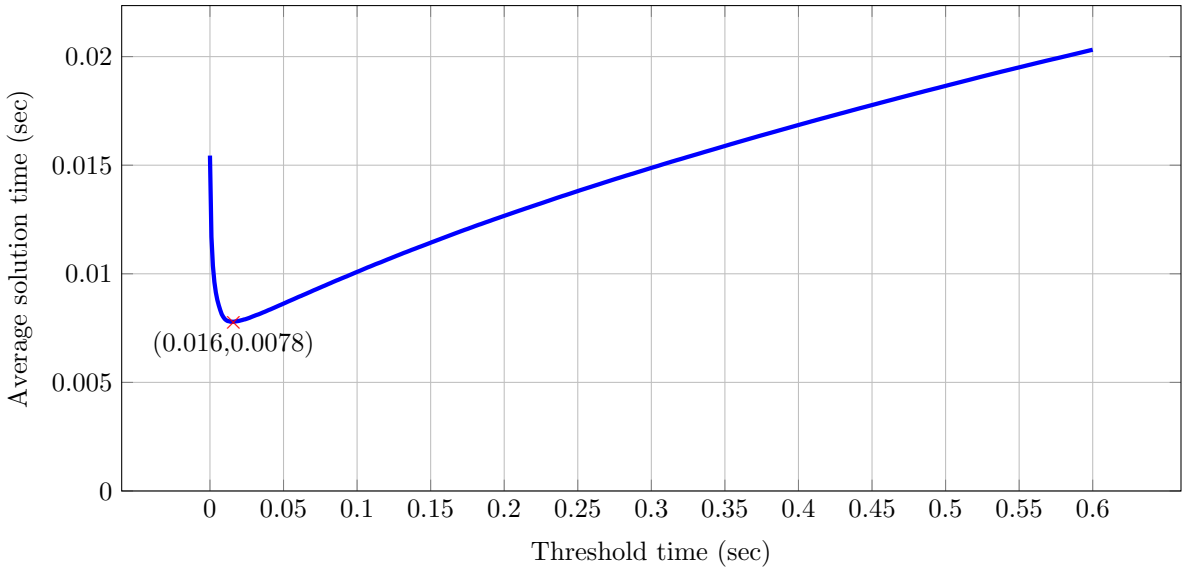

Figure 3 **Average solution time of the separation problem for varying threshold times**

seconds. Therefore, we combined the two solution techniques to benefit from their strong sides. We refer to this combination as a hybrid algorithm: we first let lbcutA algorithm run for a given time *threshold* and if the algorithm cannot identify a cut or conclude that none exists, we switch to lbcutM model. Figure 3 plots the average solution time of a single separation problem for varying threshold values. For a threshold of zero, which means running only the lbcutM model, the average separation problem solution time is 0.015 seconds. If the threshold value is infinity, which means running the lbcutA algorithm only, the average solution time increases to over 0.5 seconds. A threshold value of 0.016 seconds gives to minimum average separation problem solution time of 0.0078 seconds for the considered set of separation problem instances.

4.4. Experimental Setting

Each set in C, D, E and R classes include 20, 10, five and five instances, respectively (Table 1), totaling 350 instances. Following the convention used in the related literature, we used the same hop limit for all demands for a given problem instance. For each instance, we tested $(H^p, H^b)=(H_{min}+\Delta^p, H_{min}+\Delta^p+\Delta^b)$ for all $\Delta^p, \Delta^b \in \{0,1,2\}$. In other words, there are nine different settings for each of the 350 instances shown in Table 1, totaling 3150 problem instances, 2070 of which have grid graphs and the remaining 1080 have random graphs.

In our preliminary analyses, we observed that the NF performance is much inferior to that of FF, being slower by orders of magnitude. For example, an instance that can be

solved by FF in three seconds at the root node, can only be solved by NF in 695 seconds after exploring 12743 nodes of the branch-and-bound tree. Note that, as previously discussed, FF is also theoretically stronger than NF. Therefore, in our experimental settings, we only tested FF by using three types of branch-and-cut implementations, as shown in Table 2. The first two columns provide a grouping of implementations and their names. In the first group, we investigate the performances of the three separation algorithms. In the second group, we test the effect of time limit for lbcutM. In the final group, we test the best combination. The next three columns show the heuristic algorithms. If the ε-mincut algorithm is implemented, then there is a tick in the third column. If the lbcutM-h or lbcutA-h are implemented, then the time limit is shown in the associated column. An empty entry implies that the algorithm was not implemented for the associated version of the B&C. The following two columns show the exact algorithm. In the rightmost three columns, ϵ depicts the tailing-off parameter as percentage, 'quick-exist' implementation and the time threshold to minimize the generated cut using the lbcutM model.

Table 2 Experimental settings

Group	Name	Heuristic Algorithms			Exact Alg.		Additional Settings		
		ε-mincut	lbcut-A-h(t)	lbcutM-h(t)	lbcutM	lbcutA	ϵ (%)	quick-exit	min-lbcutM(t)
1	B&C-1	✓			✓		0.1		$t = 0.5$
1	B&C-2		$t = 0.5$			✓	0.1		$t = 0.5$
1	B&C-3			$t = 0.5$	✓		0.1		$t = 0.5$
2	B&C-4	✓		$t = 0.1$	✓		0.1	✓	$t = 0.1$
2	B&C-5	✓		$t = 0.5$	✓		0.1	✓	$t = 0.5$
2	B&C-6	✓		$t = 1.0$	✓		0.1	✓	$t = 1.0$
3	B&C-7	✓	$t = 0.016$	$t = 0.1$	✓		0.01	✓	$t = 0.1$

4.5. Results for Single Edge Failures

As in previous studies on NDPVC, when reporting the results, we assign 7200 seconds to those instances exceeding this time limit and an optimality gap of 100% if no feasible solution could be found within the time limit. All the results reported in this study are available as an online supplement. The summary of results for single edge failures are presented in Table 3. Each line in the table corresponds to 3150 problem instances. The first two columns are the grouping and the names of implementations. The remaining set

of columns reports the average run time in seconds ('avgTime'), the average percentage optimality gap ('avgGap'), the number of instances solved ('nSolved'), and the average number of cuts generated per instance ('nCuts').

Table 3 Results for single edge failures

Group	Name	avgTime (s)	avgGap (%)	nSolved	nCuts
1	B&C-1	903.97	1.51%	2846	1595.35
1	B&C-2	1085.82	2.04%	2829	1662.40
1	B&C-3	714.02	1.53%	2967	981.39
2	B&C-4	582.44	1.18%	2974	1259.70
2	B&C-5	634.76	1.28%	2968	1122.57
2	B&C-6	728.78	1.65%	2973	1061.45
3	B&C-7	569.36	1.37%	3008	1269.20

Our first observation from Table 3 is that the minimum number of solved problem instances is 2829 by B&C-2 and the maximum number is 3008 by B&C-7. Looking at the three implementations in the first group, we observe that the lbcutM-h model performs better than the ε-mincut and lbcutA-h algorithms. A total of 2967 instances are solved, which is the best of the first three implementations. Furthermore, the average solution time is also the minimum of the first group. The main reason for this result is the following: as previously discussed, the lbcutA-h algorithm is efficient, but it gets stuck at a small fraction of instances, resulting in inefficiencies. This adversely affects the overall solution process. The implementation with ε-mincut, on the other hand, generates cuts faster than the lbcutM-h heuristic. However, when it cannot identify a cut and there exists a length bounded cut, it then needs to resort to the costly lbcutM. Therefore, in the second group, we tested the ε-mincut and the lbcutM-h models running in tandem with 0.1, 0.5 and 1.0 second of time limit for the lbcutM-h. The results of the second group show that applying the two heuristics together improves the solution times and the number of solved instances. Furthermore, adding the quick-exit feature also improves the solution process. The best performing configuration is obtained when lbcutM-h runs for 0.1 second, yielding the minimum average run time, the minimum average gap and the maximum number of instances solved. Finally, in the third group, we tested the hybrid heuristic. We also

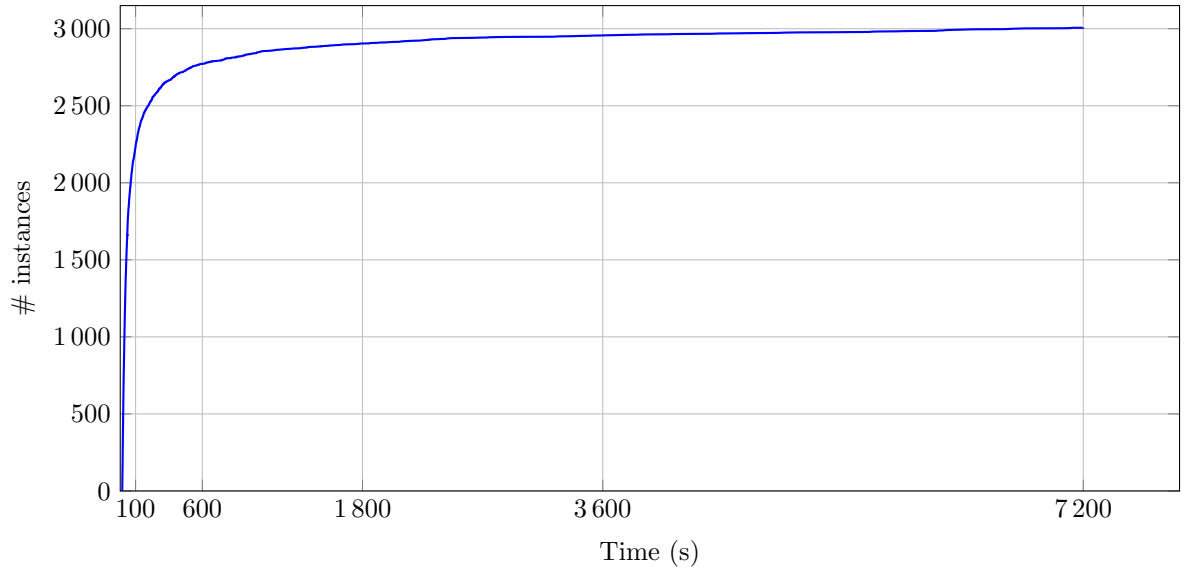

Figure 4 **Number of instances solved to optimality by B&C-7 at every time step**

reduced the tailing-off parameter from 0.1% to 0.01%, which increased the number of solved instances from 2974 in B&C-4 to 3008 in B&C-7. After determining the best parameters and settings, the following tests were carried out using B&C-7 implementation.

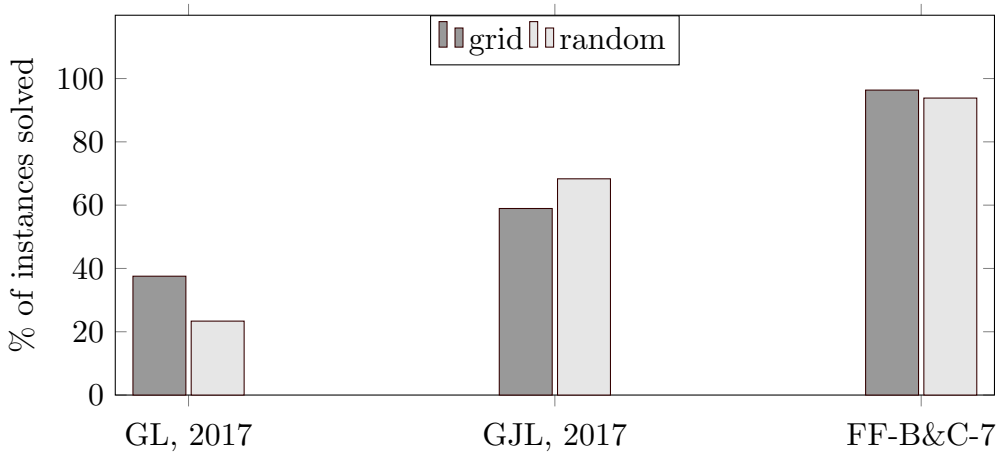

Figure 5 **Percentage of the 3150 total instances solved by Gouveia and Leitner (2017a), Gouveia, Joyce-Moniz, and Leitner (2018) and our FF-B&C-7**

Figure 4 plots the number of instances solved versus time for B&C-7. In the first 100 seconds, our formulation could solve 2098 instances. Overall, 2500 instances were solved in 230 seconds, and 2750 instances were solved in 600 seconds. Within the given 7200 seconds time limit, our model could solve 3008 instances. In Gouveia and Leitner (2017a), which

we abbreviate as GL, the authors report that out of 2070 grid instances and 1080 random instances, their best model H3 solves 781 of the grid instances and 250 of the random instances. This amounts to 1031 instances in total. In Gouveia, Joyce-Moniz, and Leitner (2018), which we abbreviate as GJL, the authors solved 1226 of the grid instances and 717 of the random instances, totaling 1943 instances (Figure 5), using their best implementation (BC3). We therefore increased the number of solved instances from the previous record of 1943 to 3008. Even though the computers on which the experiments were carried out are not the same, the difference in the number of solved instances is significant between our implementations and the state-of-the-art algorithms. One of the main reasons for this result is the tighter linear programming (LP) relaxations we obtain by FF. To better illustrate this point, we compare the LP relaxations of the FF to the tightest formulation in Gouveia and Leitner (2017a), which is H_3, on 2352 feasible instances. The average optimality gap of the LP relaxations are 2.48% and 19.09% for FF and H_3, respectively. However, there is no strict dominance of one formulation over the other, the H_3 formulation provides a better bound on 34 of the considered instances.

Table 4 Performance of separation algorithms on B&C-7

Algorithm	avgTimePerIns (s)	avg#CutPerIns
lbcut3	0.01	7.83
ε-MinCut	3.94	348.45
lbcutA-h	66.52	425.06
lbcutM-h	130.66	487.76
lbcutM	86.74	0.11
Total	287.88	1269.20

Table 4 shows the performance of the separation algorithms on our B&C-7 implementation. The first column indicates the used separation algorithm, the second column shows the average run time per NDPVC instance in seconds and the last column indicates the average number of violated cuts generated per NDPVC instance. The lbcut3 algorithm is only implemented for hop bounds of at most three. Therefore, the number of violated cuts generated is small when averaged over all instances. The main bulk of the cuts, more than 99%, are generated by the three heuristic algorithms, ε-MinCut, lbcutA-h and lbcutM-h. The fastest heuristic is the ε-MinCut. In B&C-7 implementation, the lbcutA-h and

lbcutM-h heuristics run at most for 0.016 and 0.1 seconds, respectively. The lbcutM is the last resort if all of the heuristics fail to find a violated cut. It is mainly used for proof of optimality. In very rare cases, it can identify a violated cut too. On average, 0.11 violation is identified per instance, meaning that, a violated cut is identified by the lbcutM in every 8.96 instance. The average solution time is 569.36 seconds (Table 3) and the average separation time is 287.88 seconds, which accounts to 50.56% of the average solution time.

Table 5 depicts the results of BC3 reported by Gouveia et al. (2018) and our B&C-7, grouped by sets. Our model solves all instances in the majority of the sets. Those that we could not solve completely are the largest graphs in the same letter group with the largest number of OD pairs. The average runtime and the average optimality gap are also improved over the BC3 model. Table 6 reports the same results grouped by hop limits. In all letter groups, the instances with larger hop bounds are harder to solve. However, the performance of our implementation for larger hop bounds is better than that of the BC3 model. The main reason is that the previous model of Gouveia et al. (2018) the variables have an index associated with the hop bound. Therefore, increasing hop bounds significantly increases the number of variables. In contrast, increasing the hop bound does not affect the number of variables in the FF model. To test the efficiency of our model as a function of the hop bound, we solved the case with hop limits of $\Delta^p = \Delta^b = 5$ implying $H^p = H_{min} + 5$ and $H^b = H_{min} + 10$. The results are reported in Table 7. We were able to solve 317 of the 350 instances, over 90%, and the average solution time and average gap with respect to previous results were not significantly affected. Note that the hop bounds are more relaxed with respect to the previous experiments, however they still affect the optimal solutions. When compared to the optimal costs of the network design with two disjoint paths without hop bounds, the costs are still higher in 241 of the 350 instances.

4.6. Results for Multi-Edge Failures

Another strong side of our formulation is its flexibility for solving instances in which $k \geq 3$. We tested or model for $k = 2, \ldots, 6$, and the results are reported in Table 8. The first three columns report k, the set name and the number of instances in the set. The following four columns show the number of solved instances, number of infeasible instances, average time in seconds and average gap as percentage in the given order. Our model can solve 95.49%, 92.25%, 91.08%, 90.41%, and 96.19% of the total 3150 instances for $k = 2, \ldots, 6$, respectively. The average run time is 693 seconds and the average gap is 2.28%, which

Table 5 Comparison of Flexible model with B&C-7 implementation, summarized by category

		(Gouveia et al. 2018) - BC3			FM (B&C-7)		
Set	#	# solved	avgTime (s)	avgGap (%)	# solved	avgTime (s)	avgGap (%)
C-1	180	175	522	1	180	17.2	0.00
C-2	180	121	2854	11	180	86.0	0.00
C-3	180	165	916	4	180	17.2	0.00
C-4	180	120	2746	12	180	42.0	0.00
C-5	180	39	5756	48	169	765.2	0.41
C-6	180	164	845	3	180	19.7	0.00
C-7	180	108	3240	18	180	48.8	0.00
C-8	180	60	4951	43	180	138.6	0.00
C-9	180	21	6465	67	164	1020.7	0.56
D-1	90	90	6	0	90	7.0	0.00
D-2	90	90	144	0	90	21.0	0.00
D-3	90	50	4057	35	90	119.4	0.00
D-4	90	22	6087	70	81	1588.7	1.33
D-5	90	1	7120	99	60	3421.7	17.44
E-1	45	45	110	0	45	7.6	0.00
E-2	45	40	1599	4	45	26.1	0.00
E-3	45	44	470	0	45	20.9	0.00
E-4	45	21	4383	24	45	240.7	0.00
E-5	45	41	1315	2	45	25.2	0.00
E-6	45	10	5759	67	45	245.2	0.00
E-7	45	32	2894	21	45	64.8	0.00
E-8	45	15	5211	57	38	2130.5	1.11
E-9	45	36	2227	10	45	42.5	0.00
E-10	45	9	6028	76	44	1141.7	0.05
E-11	45	23	4074	39	45	209.8	0.00
E-12	45	9	6050	75	20	4623.3	32.91
R-1	45	45	64	0	45	16.3	0.00
R-2	45	38	1969	2	45	106.5	0.00
R-3	45	45	92	0	45	29.1	0.00
R-4	45	27	3404	12	45	689.4	0.00
R-5	45	45	173	0	45	26.7	0.00
R-6	45	22	4172	22	45	932.4	0.00
R-7	45	42	935	3	45	77.9	0.00
R-8	45	20	4557	35	30	3226.8	6.73
R-9	45	40	1314	8	45	53.8	0.00
R-10	45	14	5296	46	34	3083.6	3.89
R-11	45	42	1027	3	45	85.8	0.00
R-12	45	12	5508	52	28	3812.2	9.70
Total/Avg	3150	1943	3158	29	3008	569.4	1.37

Table 6 Comparison of Flexible Model with B&C-7 implementation summarized by hop-limit

Set	(Δ^p, Δ^b)	#	(Gouveia et al. 2018) - BC3			FM (B&C-7)		
			# solved	avgTime (s)	avgGap (%)	# solved	avgTime (s)	avgGap (%)
C	(0,0)	180	151	1272	7	180	23.3	0.00
	(0,1)	180	120	2687	17	180	81.4	0.00
	(0,2)	180	89	4250	30	178	231.4	0.02
	(1,0)	180	141	1726	10	180	91.2	0.00
	(1,1)	180	107	3283	23	176	236.3	0.05
	(1,2)	180	68	4951	42	174	368.2	0.19
	(2,0)	180	136	2026	14	179	202.8	0.01
	(2,1)	180	93	3750	27	174	368.2	0.22
	(2,2)	180	68	5191	48	172	552.4	0.48
D	(0,0)	50	40	2734	29	50	201.2	0.00
	(0,1)	50	29	4665	54	49	335.8	1.06
	(0,2)	50	22	6051	77	48	565.4	0.63
	(1,0)	50	35	3133	34	47	800.3	4.29
	(1,1)	50	26	5166	63	45	1150.2	3.76
	(1,2)	50	22	6034	80	44	1156.4	2.61
	(2,0)	50	34	3528	39	45	1441.5	8.40
	(2,1)	50	23	5572	71	41	1765.7	8.32
	(2,2)	50	22	6704	90	42	1867.4	4.70
E	(0,0)	60	58	709	7	60	42.9	0.00
	(0,1)	60	42	4094	38	58	363.9	0.16
	(0,2)	60	25	6537	70	59	351.6	0.05
	(1,0)	60	50	2060	21	58	547.9	3.33
	(1,1)	60	34	4873	48	55	886.6	1.99
	(1,2)	60	20	6980	75	54	992.9	1.98
	(2,0)	60	49	2555	27	55	1061.0	6.87
	(2,1)	60	30	5258	55	54	1215.8	6.97
	(2,2)	60	17	7104	80	54	1120.9	4.18
R	(0,0)	60	55	748	2	60	14.5	0.00
	(0,1)	60	41	2676	15	60	221.2	0.00
	(0,2)	60	29	4898	37	60	154.4	0.00
	(1,0)	60	58	409	0	59	574.5	0.37
	(1,1)	60	39	2826	18	51	1611.7	3.27
	(1,2)	60	29	4420	37	55	1068.6	0.92
	(2,0)	60	60	114	0	51	1757.7	4.02
	(2,1)	60	47	2367	10	49	1986.2	4.33
	(2,2)	60	34	4684	39	52	1716.6	2.32
Total/Avg		3150	1943	3158	29	3008	569.4	1.37

Table 7 Results of Flexible Model with B&C-7 implementation for (Δ_H^p, Δ_H^b)=(5,5)

Set	#	# solved	avgTime (s)	avgGap (%)
C	180	156	1253.9	3.24
D	50	42	1677.5	9.58
E	60	60	333.3	0.00
R	60	59	654.3	0.10
Total/Avg	350	317	1053.8	3.05

are insignificantly affected by increasing k. Note that increasing k changes not only the right-hand side of Constraints (5) but also the optimal objective function value. Therefore, depending on instance, the LP relaxations might be tighter or weaker for increasing k. Note that increasing k obviously yields more infeasible instances. Figure 6 plots the percentage of infeasible instances for $k = 2, \ldots, 6$. When $k = 6$, only 16.67% of the instances are feasible. Figure 7 plots the average cost of network design for the 382 instances which we found the optimal solution for all $k \leq 6$. Even though the vertical axis has no monetary measure, we observe a strong relationship between the cost and k. Each unit increase in k implies around 200 units increase in cost, which is half the design cost for $k = 2$.

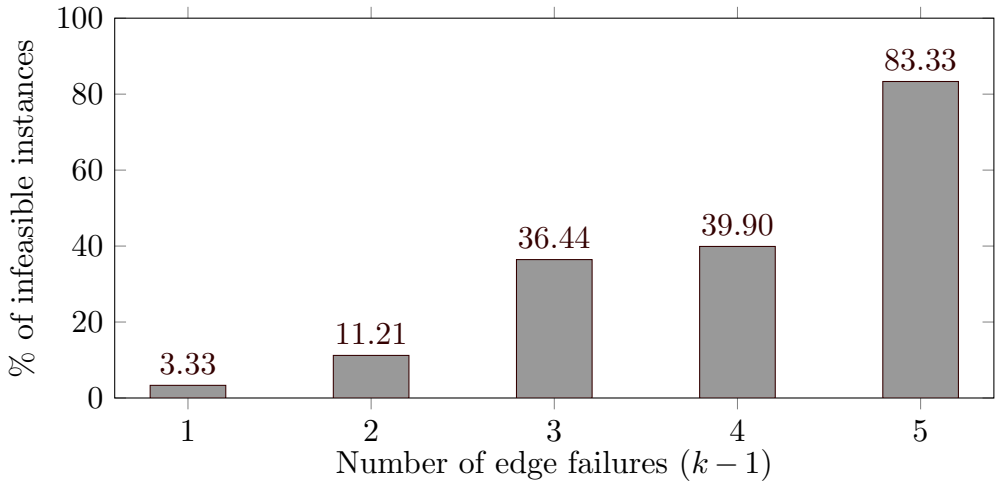

Figure 6 Percentage of infeasible solutions for varying number of edge failures

4.7. Comparison to the k-HSNDP

Similar to previous studies, we compare the results of NDPVC with those of the k-HSNDP. We recall that the k-HSNDP requires the primal and backup paths to be edge-disjoint. The

Table 8 Results of Flexible Model with B&C-7 implementation for $k = 2, \ldots, 6$

k	Set	#	# solved	# inf.	avgTime (s)	avgGap (%)
2	C	1620	1593	0	239.5	0.11
	D	450	411	4	1031.5	3.75
	E	540	507	70	731.5	2.84
	R	540	497	31	1011.7	1.69
3	C	1620	1545	49	480.9	0.27
	D	450	382	20	1509.0	9.08
	E	540	488	167	1008.3	4.02
	R	540	491	117	1041.6	1.38
4	C	1620	1529	500	532.9	0.25
	D	450	365	172	1761.7	12.66
	E	540	492	262	888.6	4.58
	R	540	483	214	1087.7	1.41
5	C	1620	1497	500	693.1	0.21
	D	450	355	172	1912.4	16.31
	E	540	498	312	745.1	2.34
	R	540	498	273	782.0	1.04
6	C	1620	1620	1542	6.2	0.00
	D	450	408	408	676.5	8.78
	E	540	498	362	658.1	1.66
	R	540	504	313	658.7	0.83
Total/Avg		15750	14661	5488	693.0	2.28

k-HSNDP results are provided to us by Gouveia et al. (2018). Out of 3150 instances, there are 2494 optimal, 106 infeasible, 451 feasible and 99 unknown instances. Using the upper bounds we obtained by FF, we compare 2551 instances of NDPVC with the k-HSNDP, for which both results are known. The results summarized by category and by hop-limit are shown in Tables 9 and 10, respectively. There are at least 1080 instances for which the cost of network design by the NDPVC is strictly less expensive than that of the k-HSNDP. Equal costs are observed in 1470 instances and one instance is shown to be feasible in

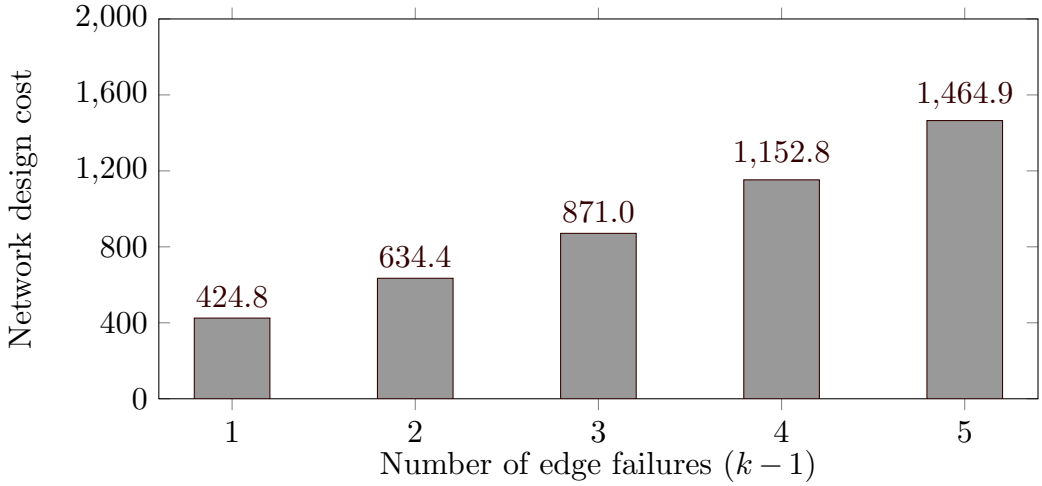

Figure 7 **Average network design cost (objective function) to hedge against varying number of edge failures**

NDPVC whereas it is infeasible in k-HSNDP. Note that we correct a few misreported numbers in the results by Gouveia et al. (2018).

5. Conclusions

We have developed a flexible natural model for the network design problem with vulnerability constraints (NDPVC), introduced by Gouveia and Leitner (2017a). This problem constitutes a new facet of the survivable network design problem by accounting for the fact that the classical network design problem with disjoint paths is a conservative approach to survivability. In the new problem definition, the concept of vulnerability is described as as hedging against a given number of 'failing edges'. In their solution approach for single edge failures, Gouveia and Leitner developed a flow-based model and used flow conservation constraints together with a 'layered' network construct to solve the problem. Gouveia et al. (2018) improve on the previous results by developing branch-and-cut algorithm. We have introduced an alternative model based on the notion of length-bounded cuts, which is more flexible for the problem with a general number of edge failures. Using only the natural variables, our implementation scales efficiently on larger networks. We can solve more than 95% of the benchmark instances for single-edge failures within a two-hour time limit. For larger numbers of edge failures, our model solves over 90% of the instances within the same time limit.

Acknowledgments

We gratefully acknowledge funding provided by the Canadian Natural Sciences and Engineering Research Council under grants 436014-2013 and 2015-06189 and by the Fonds de recherche du Québec-Nature et

Table 9 Comparison of the NDPVC results with those of k-HSNDP summarized by category

Set	#	bt	eq	feas
C-1	180	56	124	0
C-2	180	64	111	0
C-3	180	26	154	0
C-4	180	69	111	0
C-5	180	82	51	0
C-6	180	25	155	0
C-7	180	46	134	0
C-8	180	87	85	0
C-9	180	65	50	0
D-1	90	44	45	0
D-2	90	58	30	0
D-3	90	67	23	0
D-4	90	58	2	0
D-5	90	56	5	0
E-1	45	7	29	0
E-2	45	15	9	0
E-3	45	24	15	0
E-4	45	18	1	0
E-5	45	22	12	0
E-6	45	19	2	1
E-7	45	12	32	0
E-8	45	8	3	0
E-9	45	26	14	0
E-10	45	8	1	0
E-11	45	15	20	0
E-12	45	4	1	0
R-1	45	9	32	0
R-2	45	23	10	0
R-3	45	7	36	0
R-4	45	7	11	0
R-5	45	10	32	0
R-6	45	7	6	0
R-7	45	8	33	0
R-8	45	5	9	0
R-9	45	8	33	0
R-10	45	6	5	0
R-11	45	4	38	0
R-12	45	5	6	0
Total	3150	1080	1470	1

Table 10 Comparison of the NDPVC results with those of k-HSNDP summarized by hop-limit

Set	(Δ^p, Δ^b)	#	bt	eq	feas
C	(0,0)	180	23	157	0
	(0,1)	180	111	67	0
	(0,2)	180	120	55	0
	(1,0)	180	17	153	0
	(1,1)	180	60	107	0
	(1,2)	180	85	77	0
	(2,0)	180	12	148	0
	(2,1)	180	40	114	0
	(2,2)	180	52	97	0
D	(0,0)	50	16	29	0
	(0,1)	50	48	0	0
	(0,2)	50	46	2	0
	(1,0)	50	36	11	0
	(1,1)	50	38	7	0
	(1,2)	50	40	5	0
	(2,0)	50	18	20	0
	(2,1)	50	20	15	0
	(2,2)	50	21	16	0
E	(0,0)	60	4	14	0
	(0,1)	60	43	3	1
	(0,2)	60	40	6	0
	(1,0)	60	20	16	0
	(1,1)	60	19	16	0
	(1,2)	60	20	16	0
	(2,0)	60	10	23	0
	(2,1)	60	11	21	0
	(2,2)	60	11	24	0
R	(0,0)	60	1	30	0
	(0,1)	60	26	28	0
	(0,2)	60	27	28	0
	(1,0)	60	4	41	0
	(1,1)	60	13	22	0
	(1,2)	60	11	24	0
	(2,0)	60	7	25	0
	(2,1)	60	4	26	0
	(2,2)	60	6	27	0
	Total	3150	1080	1470	1

technologies under grant NC-198837. We thank the Associate Editor and three referees for their constructive comments which have helped improve the quality of this paper. We also thank Luis Gouveia, Martim Joyce-Moniz and Markus Leitner for kindly sharing the optimal solutions and the LP relaxations of their models and the k-HSNDP with us.

References

Adámek J, Koubek V (1971) Remarks on flows in network with short paths. *Commentationes Mathematicae Universitatis Carolinae* 12(4):661–667.

Arslan O, Karaşan OE, Majhoub AR, Yaman H (2017) A branch-and-cut approach for the alternative fuel refueling station location problem with routing. *Manuscript under review.*

Baier G, Erlebach T, Hall A, Köhler E, Schilling H, Skutella M (2006) Length-bounded cuts and flows. *Automata, Languages and Programming* 679–690.

Balakrishnan A, Altinkemer K (1992) Using a hop-constrained model to generate alternative communication network design. *ORSA Journal on Computing* 4(2):192–205.

Bendali F, Diarrassouba I, Mahjoub AR, Didi Biha M, Mailfert J (2010) A branch-and-cut algorithm for the k-edge connected subgraph problem. *Networks* 55(1):13–32.

Botton Q, Fortz B, Gouveia L, Poss M (2013) Benders decomposition for the hop-constrained survivable network design problem. *INFORMS Journal on Computing* 25(1):13–26.

Dahl G, Gouveia L, Requejo C (2006) On formulations and methods for the hop-constrained minimum spanning tree problem. Resende MGC, Pardalos PM, eds., *Handbook of Optimization in Telecommunications*, 493–515 (Boston, MA: Springer US).

Ford LR, Fulkerson DR (1956) Maximal flow through a network. *Canadian Journal of Mathematics* 8(3):399–404.

Fortz B, Labbé M (2002) Polyhedral results for two-connected networks with bounded rings. *Mathematical Programming* 93(1):27–54.

Fortz B, Labbé M, Maffioli F (2000) Solving the two-connected network with bounded meshes problem. *Operations Research* 48(6):866–877.

Fortz B, Mahjoub AR, McCormick ST, Pesneau P (2006) Two-edge connected subgraphs with bounded rings: Polyhedral results and branch-and-cut. *Mathematical Programming* 105(1):85–111.

Golovach PA, Thilikos DM (2011) Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discrete Optimization* 8(1):72–86.

Gouveia L (1996) Multicommodity flow models for spanning trees with hop constraints. *European Journal of Operational Research* 95(1):178–190.

Gouveia L (1998) Using variable redefinition for computing lower bounds for minimum spanning and Steiner trees with hop constraints. *INFORMS Journal on Computing* 10(2):180–188.

Gouveia L, Joyce-Moniz M, Leitner M (2018) Branch-and-cut methods for the network design problem with vulnerability constraints. *Computers & Operations Research* 91:190 – 208.

Gouveia L, Leitner M (2017a) Design of survivable networks with vulnerability constraints. *European Journal of Operational Research* 258(1):89–103.

Gouveia L, Leitner M (2017b) Online data repository. URL `http://homepage.univie.ac.at/markus.leitner/research/instances/NDPVC-instances.tar.gz`.

Grötschel M, Monma CL, Stoer M (1995) Design of survivable networks. Ball M, Magnanti T, Monma C, Nemhauser G, eds., *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, 617–672 (Elsevier, Amsterdam).

Kerivin H, Mahjoub AR (2005) Design of survivable networks: A survey. *Networks* 46(1):1–21.

Koch T, Martin A (1998) Solving steiner tree problems in graphs to optimality. *Networks* 32(3):207–232.

Lovász L, Neumann-Lara V, Plummer M (1978) Mengerian theorems for paths of bounded length. *Periodica Mathematica Hungarica* 9(4):269–276.

Mahjoub AR, McCormick ST (2010) Max flow and min cut with bounded-length paths: complexity, algorithms, and approximation. *Mathematical Programming* 124(1-2):271–284.

Menger K (1927) Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae* 10(1):96–115.

Savelsbergh MWP (1994) Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing* 6(4):445–454.

Sherali HD, Driscoll PJ (2000) Evolution and state-of-the-art in integer programming. *Journal of Computational and Applied Mathematics* 124(1):319–340, Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

Stoer M (1992) Design of survivable networks, volume 1531, Lecture Notes in Mathematics, Springer, Berlin Heidelberg.